Normal Equation, Data Fitting, and QR

1 Approximate Solutions of a Matrix Equation.

For the equations A$\mathbf{x} = \mathbf{b}$ below, write the normal equation and then use it to find the best approximate solution $\hat{\mathbf{x}}$. For extra understanding, check if $\hat{\mathbf{x}}$ is actually also a solution to the original equation.

(a) $\begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$

(b) $\begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} 1 \\ 8 \end{bmatrix}$

(c) $\begin{bmatrix} 1 & -2 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \end{bmatrix}$

(d) $\begin{bmatrix} 1 & -2 \\ 2 & -4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$

(e) $\begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

(f) $\begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \\ 0 \\ 1 \end{bmatrix}$

2 Least Squares Data Fitting Using the Normal Equation.

For the following data sets and the function $f(t)$,

(i) write a matrix equation $A\mathbf{x} = \mathbf{b}$ to compute the coefficients of $f(t)$,

(ii) write the associated normal equation $A^{\mathrm{T}}A\hat{\mathbf{x}} = A^{\mathrm{T}}\mathbf{b}$,

(iii) solve for the best approximate solution for the coefficients of $f(t)$ and write $\hat{f}(t)$

(a) $f(t) = a$

t	-1	0	1	2
$f(t)$	-3	-1	1	0

(b) $f(t) = bt$

t	-1	0	1	2
$f(t)$	-3	-1	1	0

(c) $f(t) = a + bt$

t	-1	0	1	2
$f(t)$	-3	-1	1	0

(d) $f(t) = a + bt^2$

t	-1	0	1	2
$f(t)$	-3	1	1	-1

(e) $f(t) = a + {}^{b}/_{t}$

t	$1/3$	$1/2$	1
$f(t)$	3	0	-6

(f) $f(t) = a\cos(t) + b\cos(2t)$

t	0	$\pi/2$	π	$3\pi/2$
$f(t)$	1	-1	3	2

3 Using Scaled QR-Decomposition

Use the scaled QR-decompositions given below to solve.

(a) $\begin{bmatrix} 2 & 1 & -2 \\ 1 & 2 & 2 \\ 2 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & -1 \\ 0 & 2 & 1 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -3 \\ 9 \\ 3 \end{bmatrix}$

(b) $\begin{bmatrix} 2 & 1 & 4 \\ 1 & 1 & -5 \\ -3 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 2 & -3 \\ 0 & 2 & -4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 8 \\ -2 \end{bmatrix}$

(c) $\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & -3 \\ 0 & 2 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 1 \\ -1 \end{bmatrix}$

(d) $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 2 & -2 & 0 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 0 \\ 8 \\ 5 \end{bmatrix}$

4 Computing Scaled QR-Decomposition

Compute a scaled QR decompositions for the matrices below.

(a) $\begin{bmatrix} 1 & 2 & 3 \\ 1 & 0 & -9 \\ 2 & 5 & -3 \end{bmatrix}$

(b) $\begin{bmatrix} 1 & -1 & 8 \\ 2 & -7 & 6 \\ 2 & -6 & -1 \end{bmatrix}$

(c) $\begin{bmatrix} 2 & -8 & 11 \\ 3 & 7 & -3 \\ -5 & 1 & -5 \end{bmatrix}$

(d) $\begin{bmatrix} -9 & -7 & 8 \\ 7 & 10 & -9 \\ 1 & -2 & 4 \end{bmatrix}$

(e) $\begin{bmatrix} 1 & 5 & 4 \\ 1 & -1 & -10 \\ 1 & 5 & 0 \\ 1 & 3 & -2 \end{bmatrix}$

(f) $\begin{bmatrix} 1 & 4 & 0 \\ 1 & 3 & 3 \\ 1 & 2 & 4 \\ 1 & 3 & 1 \end{bmatrix}$

(g) $\begin{bmatrix} 1 & 5 & -7 \\ 1 & 7 & 7 \\ 2 & 7 & -5 \\ 2 & 7 & -5 \end{bmatrix}$

(h) $\begin{bmatrix} 1 & -1 & 5 \\ 2 & -7 & 1 \\ 2 & -4 & 9 \\ 2 & -8 & -6 \end{bmatrix}$

5 MatLab

- Recall that transpose in MatLab is `'`. To solve the normal equation it is fastest enter `A` and `b` first and then use the command `(A' * A) \ (A' * b)` to divide $A^{\mathrm{T}}A\hat{\mathbf{x}} = A^{\mathrm{T}}\mathbf{b}$.

  Example. Find the best approximate solution to the matrix equation $\begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 2 & 3 \\ -1 & 5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -1 \\ 7 \\ -2 \\ 3 \end{bmatrix}$

```
1 >> A = [1 -1; 1 0; 2 3; -1 5]
2 >> b = [-1; 7; -2; 3]
3 >> x_hat = (A' * A) \ (A' * b)
```

  In this case the error vector is given by

```
4 >> e = b  -  A * x_hat
```

  with squared error

```
5 >> e' * e
```

- It is slightly faster to enter column vectors into MatLab as transposes of row vectors. The same trick works for matrices that don't have many columns. For example lines 1 and 2 above could be replaced by

```
1 >> A = [1 1 2 -1; -1 0 3 5]'        %  A = [1 -1; 1 0; 2 3; -1 5]
2 >> b = [-1 7 -2 3]'                 %  b = [-1; 7; -2; 3]
```

- To solve data fitting problems using MatLab, we first enter the independent variable `t`, then we build the matrix `A`, column by column, plugging `t` into the terms multiplied by unknown constants.

  Example. Fit the formula $f(t) = a + b\,t + c\,t^2$ to the data

t	-1	0	1	2	3	4
$f(t)$	-3	1	1	-1	-6	-8

```
6 >> t = [-1 0 1 2 3 4]'              %  column vector of  t-values
7 >> A = [t.^0 t.^1 t.^2]             %  matrix of  [ 1  t  t^2 ]
8 >> f = [-3 1 1 -1 -6 -8]'           %  column vector of  f-values
9 >> coeff = (A' * A) \ (A' * f)      %  normal eqn for   A * coeff = f
```

  Line 7 uses `t.^2` instead of `t^2` to tell MatLab to square `t` **element-wise** rather than square `t` as a **matrix**. Without the `.` MatLab would have returned an error, since you cannot multiply two `6x1` matrices.

- MatLab has a command `qr` which gives the (unscaled) QR-decomposition. The (unscaled) QR decomposition differs from our version by insisting that all columns of Q have length 1. In practice this means dividing each column of `Q` by its length and multiplying the corresponding rows of `R` by this length.

  We can write MatLab code to compute scaled QR decomposition using a `for` loop to build `Q` column by column. Begin with `Q` = first column of `A`, and then convert the other columns of `A` to be columns of `Q` by computing error vectors for the approximate solution to Q$\mathbf{x}$ = (next column of A).

  Example. Compute the scaled QR decomposition of $A = \begin{bmatrix} 1 & 4 & 11 \\ 2 & 6 & -9 \\ 2 & 1 & -10 \end{bmatrix}$.

```
10 >> A = [1 4 11; 2 6 -9; 2 1 -10]      %  Enter matrix A
11 >> Q = A(:,1)                         %  Begin with Q = col 1 of A
12 >> R = [1 0 0; 0 1 0; 0 0 1]          %  Begin with R = identity matrix
13 >> for (col = 2:size(A,2))            %  Loop over columns of A
14       x_hat = (Q' * Q) \ (Q' * A(:,col)) %  Normal eqn for  Q x = (next col)
15       e = A(:,col) - Q * x_hat           %  Error vector for x_hat
16       Q = [Q e]                          %  Error vector is next col of Q
17       R(1:col,col) = x_hat               %  x_hat is next col of R
18    end                                %  End of loop
```

  Line 13 uses the command `size(A,2)` to get the number of columns of A. The command `size(A,1)` gives the number of rows of A, and the command `size(A)` gives the number of rows and columns.